文章编号：0258-0926(2014)04-0142-05; doi: 10.13832/j.jnpe.2014.04.0142

基于 RMC 的计数器数据分解方法研究

梁金刚[1]，王 侃[1]，余纲林[1]，佘 顶[2]，
柴晓明[3]，强胜龙[3]，姚 栋[3]

1. 清华大学工程物理系，北京，100084；2. 清华大学核能与新能源技术研究院，北京，100084；
3. 中国核动力研究设计院核反应堆系统设计技术重点实验室，成都，610041

摘要：蒙特卡罗模拟方法（蒙卡方法）在反应堆物理分析中的应用受计算机内存不足的限制，数据分解方法是一种有效的解决思路。对蒙卡方法的内存占用进行定量分析，并基于自主堆用蒙特卡罗程序（RMC），采取了同步式和异步式 2 种通信方法，设计并实现计数器数据分解算法；通过数值试验测试算法的性能，结果表明，计数器数据分解算法能够明显减少内存占用，而且不会对程序的并行性能产生影响。

关键词：数据分解；计数器；蒙特卡罗；内存；RMC；并行效率
中图分类号：TL32 文献标志码：A

0 引 言

在反应堆物理计算领域，求解中子输运问题的蒙特卡罗模拟方法（简称蒙卡方法）因为在处理复杂几何和复杂能谱问题方面的优势，近几年越来越受到重视。然而，将蒙卡方法应用于反应堆全堆芯的大规模计算分析还存在一些需要解决的问题，其中计算时间长、内存占用大[1]是关键问题之一。

已有的研究表明，采取高性能粒子并行算法后，蒙特卡罗程序（简称蒙卡程序）的计算时间显著减小，即具有很好的并行性能[2-3]。对于内存占用大的问题，发现相对较晚，解决方法还不成熟，是目前研究的热点。有 2 种节省内存占用的可行方法，即区域分解[4]与数据分解[5-6]算法。区域分解方法是指将计算对象从模型上划分为不同的子区域，分别由并行的处理器进行模拟计算，通过将大模型问题转化为小模型问题减小内存占用。因为在设计中需要考虑区域划分、粒子越区域通信以及相应的负载均衡、计数统计等若干问题，区域分解算法较为复杂。数据分解算法是针对蒙卡程序内存占用问题的一种新的解决方法，直接将蒙卡计算的数据划分并分配至不同的处理单元，利用并行通信实现数据的读取和更新。文献[7]对区域分解算法进行了研究。本文基于自主堆用蒙卡程序（RMC）[8]，设计并实现计数器数据分解算法。

1 蒙卡程序计算内存需求分析

中子输运蒙卡程序的数据通常可分为以下几类：几何数据、核截面数据、粒子信息数据、计数器数据以及其他临时数据和必要的支持数据。其中最后一类数据占用内存较小，且比较固定；前 4 类数据则会随着计算规模增长，需要进行考察。数据内存均以 RMC 为参考逐一进行分析。

1.1 几何数据

计算模型中几何数据的大小，既取决于计算对象的几何规模（如栅元的数量、交界面的复杂度等），也与程序对几何的建模方法有关。RMC 的几何建模方法为构建实体几何（CSG）方法，即利用曲面（SURFACE）通过组合逻辑构建栅元（CELL）。曲面在程序中由类型和特征尺寸数据描述，栅元则是曲面的逻辑组合序列，因此，可以估计几何数据的内存占用不会很大。

简单地定量估计：1 个曲面的数据包含 1 个曲面编号（整型），1 个曲面类型号（整型）和平均 5 个特征参数（双精度型）；一个栅元的数据包括平均 10 个曲面编号及相应布尔逻辑（整型），一个曲面编号（整型）和材料编号（整型），以及

收稿日期：2013-12-20；修回日期：2014-02-25

温度、体积等数据（双精度型）。

每个曲面的数据大小约(2×4+5×8)≈50 bytes（bytes 表示字节），每个栅元的数据大小约为(21×4+2×8+Δ)<200 bytes。可以看出，即使计算模型的栅元和曲面数目均达到百万量级，两者的数据大小也只是在 100 Mbytes 量级。而在实际计算中，程序还具备简化建模的功能，如层级结构和重复结构描述方法，可以大大减小对栅元和曲面的定义数目。比如，利用 RMC 模拟 Hoogenboom-Martin 全堆基准题[9]，只需要定义 26 个栅元（包含层级和重复几何栅元）和 18 个曲面，即可描述含有约 2.1×10^7 个几何单元的全堆模型，其全部几何数据的大小不足 1 Mbytes。

1.2 核截面数据

蒙卡程序在进行模拟前，一般先从外部截面数据库中将需要的核素的核截面数据读入内存。单个核素的核截面数据（特定温度下）按照特定格式（如 ACE 格式）存储，内部包含数据结构较为复杂。为便于估计，可以从外部核截面库估算单个核素平均数据的大小。以自主核截面处理程序（RXSP）加工的常用 ACE 截面库为例，它包含 393 个同位素，而存储大小为 785 Mbytes，因此，特定温度下单个核素的截面数据平均大小约为 2 Mbytes。

对于一般的稳态计算，核截面数据的内存占用是可以接受的，但是如果考虑堆芯温度分布以及核素数目随着燃耗增多，核截面数据的大小可能超过 10 Gbytes。然而，由于核素分布广泛的缘故，核截面数据不易于通过划分区域而减小尺寸。数据分解以及在线多普勒展宽是可以研究的解决方法。

1.3 粒子信息数据

使用蒙卡程序进行临界计算时，需要时刻保存新生中子的状态以进行新一代的模拟。一般保存 3 个基本的粒子信息：位置、运动方向和能量。其中位置和运动方向都是包含 3 个变量的参数。考虑到一般同时存有 2 列粒子信息数据，分别为当前代和下一代源中子。

每个中子信息的数据大小约 2×7×8=112 bytes，全部粒子信息的内存占用约为 $112N_{\mathrm{hsty}}$ bytes，其中 N_{hsty} 为每代粒子数。一般地，单处理器模拟的代粒子数不超过 10^6（采用粒子并行方法），所以粒子信息的数据大小在 Gbytes 量级以内，是可以接受的。

1.4 计数器数据

计数器用于计算中子通量、功率、反应率或核素截面等。计数器的使用与否、数目是由用户指定的（燃耗计算中必须统计燃耗栅元的单群截面数据）。为了获取一个计数参数的结果，每个计数器需要包含 5 个基本的统计变量（单次统计值、累加值、平方累加值、平均值和标准偏差），同时计数器还需要若干限定参数，如指定的计数栅元、能量区域等。综合计算，每个计数器单元的数据大小约为 50 bytes。不过，计数器数目有很大的变化范围，以普通压水堆为例，燃耗元件数目约为 $(17\times17)\times(21\times21)\approx1.27\times10^5$，如果想要统计全堆芯精细通量分布，对元件径向和轴向分别划分为 5 和 100 段，那么通量计数器的数目为 6.4×10^9，计数器的数据大小为 320 Gbytes。可见，计数器是蒙卡程序内存占用的关键部分。

1.5 内存占用分析

根据前面的分析，表 1 总结蒙卡程序各模块数据的内存占用需求。

对蒙卡程序内存需求的分析，一方面证实了蒙卡内存问题的存在性，另一方面，定量的数据尺寸分析，勾勒出内存占用的分布情况，指出计数器数据是内存问题的关键所在。以此为指导，将对计数器进行数据分解，寻找内存问题的解决办法。

表 1 蒙卡程序计算内存需求分析

Table 1 Analysis of Memory Demand in MC Simulation

数据模块	单元数据量	单元数目规模	总数据量规模
几何数据	<200 bytes	10^6	<100 Mbytes
核截面数据	约 2 Mbytes	10^3	10 Gbytes
粒子信息数据	112 bytes	10^6	1 Gbytes
计数器数据	50 bytes	10^9	≥100 Gbytes
其他	—	—	<100 Mbytes

2 计数器数据分解算法的设计及实现

2.1 计数器数据分解

由于计数器数据量大是蒙卡程序内存问题的主要原因，对计数器数据进行分解，是数据分解的优先选择，如图 1。

总体上，计数器数据分解算法包含 3 部分任务：①分解数据，将计数器分配至不同的处理器，

建立计数器编号与位置的索引关系；②数据通信，将粒子输运过程中产生的计数运送到正确的位置；③分解数据的利用。对于任务①，在 RMC 程序中，计数器数据是计数器元数据的一维向量，因而易于分解并分配到并行的处理器。即将计数器按处理器数目均分，每个处理器创建相应数量的计数器。其次，创建计数器编号与处理器的索引，用于定位某一计数器所在的目标处理器和偏移位置。对于任务②，利用消息传递并行（MPI），设计了 2 种通信机制不同的算法：同步通信和异步通信算法。任务③指后续如何有效利用分解的数据，比如输出数据或用于燃耗计算等。

计数器数据分解算法与粒子并行算法同时存在，两者相互配合。

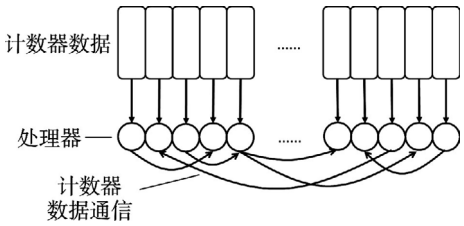

图 1 计数器数据分解示意图

Fig.1 Schematic Diagram of Tally Data Decomposition Algorithm

2.2 同步式通信算法

阻塞式通信即同步式通信，所有的处理器统一执行通信并同步。对于每个处理器，当粒子输运过程中产生了非本地计数，先保存至缓冲区，然后根据设定的通信周期启动阻塞式通信。所有处理器完成通信后继续粒子模拟，如此循环。在一代模拟结束时同样执行通信。通信周期是启动通信的条件，周期越长则缓冲区存储的计数信息越多，单次通信量变大，而总的通信次数越少。比如，可以设置通信周期为处理器模拟完 N 个粒子。

同步通信算法（阻塞式）如下：

```
decompose tally data to different processors
for cycle = 1 to CycleNum
|    for neutron = 1 to NeutronNum
|    |    tracking history
|    |    |  if(tally not local)
|    |    |    save it to tally buffer(target processor, tally position)
|    |    end one history
|    |    if(meet communication period or last neutron)
|    |    |    if(myId < targetId)
|    |    |    |    send tally number and data (MPI_Send)
|    |    |    |    receive tally number and data (MPI_Recv)
|    |    |    else if(myId > targetId)
|    |    |    |    receive tally number and data (MPI_Recv)
|    |    |    |    send tally number and data (MPI_Send)
|    |    |    add received tally to local data
|    end for loop
end for loop
```

具体在数据交换过程中，采取阻塞式通信操作（MPI_Send，MPI_Recv），为了避免处理器间出现通信死锁，根据处理器的编号大小确定发送、接收顺序。

2.3 异步式通信算法

借助于非阻塞通信操作，异步式通信算法可以实现通信与计算的重合。

异步通信算法（非阻塞式）如下：

```
decompose tally data to different processors
for cycle = 1 to CycleNum
|    post non-blocking receives from other processors(MPI_Irecv)
|    for neutron = 1 to NeutronNum
|    |    tracking history
|    |    |  if(tally not local)
|    |    |    save it to tally buffer (target processor, tally position)
|    |    |    if(buffer is full)send this buffer to target(MPI_Isend)
|    |    end one history
|    |    if(meet check period)
|    |    |    test all receives(MPI_Test)
|    |    |    while(one received success)
|    |    |    |    add received tally to local data
|    |    |    |    if(not last recv)post receive again (MPI_Irecv)
|    |    |    end while loop
|    |    if(last history)
|    |         send all buffer out(MPI_Isend)
|    |         wait all send and receive success(MPI_Wait)
|    end for loop
|    process cycle end
end for loop
```

在非阻塞式通信算法中，每个处理器时刻张贴非阻塞式接收操作，并在计数缓冲区满后即刻执行非阻塞式发送，按照一定的周期检测接收操

作的完成情况，处理接收的数据，并继续张贴接收操作。

3 数值测试

在 RMC 中，分别实现 2 种计数器数据分解算法。2 种算法均能得到正确的计数结果。实际上，数据分解算法仅改变数据的存储方式，对数据本身的变化过程并不产生影响，因而，在同样的计算条件下，使用数据分解算法的程序，得到与普通程序一致的结果。这种天然具备的可重复性，也是数据分解算法的一大优势。

为了测试算法的性能，使用 Hoogenboom-Martin 全堆基准题[9]进行数值测试，对其中的 6362400 个栅元芯块的使用中子通量计数，分别用普通版 RMC、同步和异步通信数据分解算法的 RMC 3 个程序进行计算。

首先，在单节点多核服务器上进行测试，表 2 记录在不同并行核数目下 3 个程序的执行时间和内存占用情况。其中服务器配置为 2×Intel® Xeon® CPU E5-2690@2.90GHz, Windows Server 操作系统。计算条件为 Kcode 100000 10 20，即每代模拟 10^6 历史，进行 10 个非活跃代和 10 个活跃代的模拟，为了统计数据分解算法的并行性能，只统计活跃代的计算时间。

表 2 多核服务器计数器数据分解算法的计算时间和内存占用

Table 2 Calculation Time and Average Memory of Tally Data Decomposition Algorithms on Multicore Server

程序	RMC		同步算法 RMC		异步算法 RMC	
并行核数	时间/min	内存/MB	时间/min	内存/MB	时间/min	内存/MB
1	53.8	1006.2	54.2	1006.1	56.8	1006.3
2	29.6	942.0	30.1	820.5	30.2	820.9
4	15.1	909.9	15.0	727.5	15.6	728.2
6	10.4	899.2	10.6	696.5	10.8	697.5
8	8.0	893.9	8.3	681.1	8.1	682.5
12	5.4	888.5	5.6	665.5	5.4	668.4

由表 2 可以看出，计数器数据分解可以显著节省蒙卡程序的内存占用。根据 1.4 节的定量计算，6362400 个计数器占用总内存量约为 250 Mbytes，采用计数器数据分解后程序的内存占用显著减小，变化趋势与表 2 相符。

为了测试计数器数据分解算法的扩展性能，进行了更大并行规模的计算。选取清华大学高性能计算平台——“探索 100”服务器，每代可模拟 10^7 历史，表 3 列出了 RMC、同步和异步计数器数据分解算法的计算时间和并行效率。随着并行核数的增加，异步算法保持较好的并行效率，计数器数据通信带来的效率折损在 5%以内，相比之下，同步算法在并行核数较少时并行性能很好，但随着核数的增多，并行性能显著降低。

表 3 集群平台计数器数据分解算法的并行性能

Table 3 Parallel Performance of Tally Data Decomposition Algorithms on Clusters

并行核数	RMC		同步算法 RMC		异步算法 RMC	
	时间/min	并行效率/%	时间/min	并行效率/%	时间/min	并行效率/%
12	66.24	100.0	68.18	97.2	66.24	100.0
24	33.48	98.9	34.63	95.6	33.48	98.9
48	16.81	98.5	17.56	94.3	16.81	98.5
72	11.21	98.5	11.89	92.8	11.21	98.5
96	8.43	98.2	9.06	91.4	8.43	98.2
120	6.78	97.7	7.40	89.6	6.78	97.7
144	5.62	98.2	6.49	85.1	5.62	98.2
240	3.39	97.7	4.60	72.1	3.39	97.7
360	2.28	96.9	4.10	53.9	2.28	96.9

4 总 结

对蒙卡程序计算内存需求进行了定量分析，明晰了各模块数据的尺寸，确定计数器是蒙卡内存问题的主要部分。基于 RMC，设计并实现了计数器数据分解算法，包括同步式和异步式 2 种通信机制。对计数器数据分解算法进行数值测试，分析了两种算法在不同的处理器并行计算情况下的性能。结果表明，计数器数据分解能够节省计数器数据的内存占用，且 2 种通信算法不会显著影响程序并行效率。

参考文献：

[1] Martin W R. Challenges and prospects for whole-core Monte Carlo analysis[J]. Nuclear Engineering and Technology, 2012, 44(2): 151-160.

[2] Romano P K, Forget B. Parallel fission bank algorithms in Monte Carlo criticality calculations[J]. Nuclear Science and Engineering, 2012,170: 125-135

[3] Brantley P S. Recent advances in the mercury Monte Carlo particle transport code[C]. International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering. 2013.

[4] Urbatsch T J, Evans T M. Parallel implicit Monte Carlo in C++[C]. International Symposium on Computing in Object Oriented Parallel Environments. 1998.

[5] Brown F B. Recent advances and future prospects for Monte Carlo[J]. Progress in Nuclear Science and Technology, 2011, 2:1-4.

[6] Romano P K, Forget B, Brown F. Towards scalable parallelism in Monte Carlo transport codes using remote memory access[J]. Progress in Nuclear Science and

Technology, 2011, 2:670–675.
[7] 梁金刚，蔡云，王侃，孙嘉龙. 中子输运蒙特卡罗模拟的区域分解方法研究[C]. 核反应堆系统设计技术重点实验室 2012 年度学术交流会. 2012.
[8] Wang Kan, Li Zeguang, She Ding. Progress on RMC - a Monte Carlo neutron transport code for reactor analysis[C]. International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering. 2011.
[9] Hoogenboom J E, Martin W R. The Monte Carlo performance benchmark test - aims, specifications and first results[C]. International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering. 2011.

Research on Tally Data Decomposition Algorithms Based on Reactor Monte Carlo Code RMC

Liang Jingang[1], Wang Kan[1], Yu Ganglin[1], She Ding[2], Chai Xiaoming[3], Qiang Shenglong[3], Yao Dong[3]

1. Department of Engineering Physics, Tsinghua University, Beijing, 100084, China;
2. Institute of Nuclear and New Energy Technology, Tsinghua University, Beijing, 100084, China;
3. Science and Technology on Reactor System Design Technology Laboratory, Nuclear Power Institute of China, Chengdu, 610041, China

Abstract：The applications of Monte Carlo method in reactor physics analysis are restricted due to excessive memory demand in solving large-scale problems, while data decomposition is supposed to be a remedy. In this paper quantitative memory requirements in MC simulation are analyzed. Two types of tally data decomposition algorithms, which utilize synchronous and asynchronous communications, are designed and implemented in Reactor Monte Carlo code(RMC). Numerical tests are carried out to evaluate performance of new algorithms. It is shown that tally data decomposition algorithm can reduce memory size effectively while parallel efficiency of the code is not affected.

Key words：Data decomposition, Tally, Monte Carlo, Memory, RMC, Parallel efficiency

作者简介：

梁金刚（1989—），男，在读博士生。2010 年毕业于清华大学工程物理系核工程与核技术专业，获学士学位。现主要从事反应堆物理计算方法的研究工作。

王 侃（1965—），男，教授，博士生导师。1993 年毕业于西安交通大学能源与动力工程系反应堆工程与安全专业，获博士学位。现从事核反应堆物理数值计算、多物理耦合以及先进和新概念核能系统的研究和教学工作。

余纲林（1977—），男，助理研究员。2007 年毕业于清华大学工程物理系核科学与技术专业，获博士学位。现主要从事反应堆物理方面的研究和教学工作。

（责任编辑：张祚豪）